# Drag'n Go: simple and fast navigation in virtual environment

Category: Research

Figure 1: Drag'n Go in various situations.


## ABSTRACT

In this paper we introduce the *Drag'n Go* technique to navigate in multi-scale virtual 3d environment. This new technique takes its root from the point of interest (POI) [10] approach where the user selects a target to reach. The biggest difference between the two is that with Drag'n Go the user keeps full control of its position relative to the target as well as its traveling speed. The technique requires only a 2d input and consequently, can be used with a large amount of devices like mouse, touch or pen screen. We conducted preliminary experiment that highlights that Drag'n Go is an efficient and appreciated method for touch-based device and a competitive approach for mouse-based device.



**Index Terms:** I.3.6 [Computing Methodologies]: Computer Graphics—Methodology and Techniques;
H.5.2 [Information Systems]: Information Interfaces and presentation— User Interfaces


## 1 INTRODUCTION

Navigation is one of the elementary tasks of 3d virtual environment. It is composed of two parts: locomotion where there is a physical control of the camera and the wayfinding where a path is found through the environment. Despite being widely studied, there is still need for more efficient and simple techniques. In the meantime there is an increasing number of user applications based on 3d environment and thus the demand for such navigation technique is high. We would like navigation to be intuitive and easy to use as well as efficient, allowing the user to achieve his task in short time.

In this article we introduce the *Drag'n Go* locomotion technique that exhibits a nice compromise between all these desired properties. It combines the advantages of point of interest [10] with features of direct manipulation [14]: reversibility of movement, no delay between interaction and result. Drag'n Go is based on a traveling path between the avatar position and the target position. Standard POI approaches offer limited and indirect control of the advancement speed, Drag'n Go allows users to control avatar progression along the path with simple movements on the interaction area (e.g. screen in the case of multi-touch displays).

To highlight the advantages of directness and POI approach of Drag'n Go, we carried out a pilot experiment where three navigation types were set against Drag'n Go: POI approach, a 3d navigation from game and a touch navigation technique.

The paper is organized as follows: after presenting related works on virtual navigation, we describe our technique in section 3. In section 4, we present the pilot experiment we designed to test Drag'n Go. In section 5, we discuss problems related to our technique. Finally, in section 6, we conclude and give possible directions for future work.

## 2 PREVIOUS WORKS

Locomotion in a virtual environment is the task of controlling the degrees of freedom (DoF) of the camera within the 3d scene. In most cases only six *extrinsic* camera parameters (translation and rotation, as opposed to *intrinsic* parameters such as pixel resolution, and lense) are controlled during interaction. We refer the reader to [4] for a good overview of existing navigation techniques. We can find a large amount of approaches in literature, that we categorize into two classes: approaches that map the device input to the camera parameters (either using position or speed), and approaches based on gesture recognition systems. The techniques may either be*context-sensitive* or *context-insensitive*, depending on whether or not they take advantage of the scene content to improve interaction. They may also be *direct* or *indirect*, following Shneiderman's approach [14]. Directness on its spatial aspect is closely connected to the *world in hand* technique introduced by [17].

Approaches based on simple mapping between the user input DoFs and the camera DoFs are the most popular ones. They implement real navigating objects like the flying vehicle or the walking metaphor [17] and do not yield significant latency between user action and application reaction. Such techniques are also context insensitive and are controlled either by position or speed. An option to improve navigation speed is to use hybrid control as in RubberEdge [2], an alternative solution is to use context sensitive techniques as the Point-of-Interest (POI for the remainder of this article)[10]. With POI, the user specifies a target and selects whether the avatar is going forward or backward. A logarithmic speed function connects the camera traveling speed with its distance to the target. The result is a fast movement when far away from the target, and a slow movement when close. A slightly similar approach is proposed in Ware et al [16], in which distance between camera and target specifies the camera movement speed of a flying movement. Tan [15] extends POI technique by including information about the user and the environment to specify final position and orientation according to the target. He couples a flying metaphor to move in direction of the target and an orbit to spec-

ify final position and orientation. Based on a different calculus but with very similar results to POI are camera positioning techniques that use screenspace constraints as presented in [6] and improved in [13]. A recent work done in [11] also addresses the issue of fast and precise navigation in multi-scale 3d environment. This approach shares its goals with our method but with a much less direct set of controls. Other context-sensitive approaches exist like Hovercam [9] or Stylecam [1] but they seem designed for more specific tasks than navigating in virtual environments.

Gesture-based approaches can also be a very natural way to navigate in 3d environment but they also induce delay, due to duration of the gesture realization. Among the gesture based techniques, in [8] the user draws a trajectory and at the end of the interaction, this trajectory is used for navigation. More recently, [7] introduces an interaction widget called *Navidget* defining precisely the target position and camera orientation. In addition, [7] uses a sketch-based interface to interact with the camera and during the navigation, the camera moves on a trajectory while respecting a constant duration of 100 key-frames (c.a. 3 seconds).

## 3 Drag'n Go navigation technique

Drag'n Go provides direct control of the avatar's progression using a walking path built at the beginning of the interaction. The user controls the progression of the navigation by the position of his cursor using the line between its initial position and the bottom border of the screen as a slider.


Figure 2: Example of Drag'n Go navigation, the percentage widget is for illustration purposes only.

### 3.1 Design space

Drag'n Go is based on a screen space design, i.e. the cursor (a touch, a mouse, a pen) should move in the same direction as the on-screen optical flow. This choice provides a very good stimulus-response compatibility and gives a strong sense of directness in the interaction. Some possible mappings between cursor movements and possible camera movements are given in table 1. However, such a screen-space guideline is incomplete, as the cursor movement may map to several camera movements sharing close optical flow. We carried out an experiment in which we tested different mappings between the input DoFs and the cameras DoFs. The experiment led us to the conclusion that for best performance and user satisfaction, moving forward/backward should be combined with turning left/right as primary mode while tilt and sidestepping should be a secondary mode (see table 2).

| Camera movements | Optical Flow field |
|---|---|
| Tilt up/down, Move up/down | ↓/↑ |
| Turn left/right, Sidestep left/right | ←/→ |
| Move forward/backward, Zoom in/out | ↓/↑ and ←/→ |

Table 1: Relations between camera movements and resulting optical flow.

| Cursor Mvt | Primary Camera Mvt | Alternative Camera Mvt |
|---|---|---|
| ↑ | Move backward | Tilt down |
| ↓ | Move forward | Tilt up |
| ← | Pan on right | Sidestep right |
| → | Pan on left | Sidestep left |

Table 2: Relations between cursor and camera movements with Drag'n Go .

### 3.2 Implementation

The implementation of Drag'n Go is, from a complexity point of view, comparable to that of the POI technique. With Drag'n Go the control of the avatar's progression is done through a walking path by mapping the progression along the path with the edge passing by the cursor and reaching the border of the screen.

Drag'n Go is decomposed in three steps:

- on cursor down (activation): from the cursor screen position $Ci$ cast a ray to identify the target destination $T$ in world space. Then, compute a path between the target destination $T$ and the current position of the avatar $A$.
- on cursor move (interaction): for the $y$ component, move the avatar position along the path using the current cursor position $Cc$ relative to its initial position $Ci$. The ratio $(Ci_y - Cs_y)/Ci_x$ parameterizes the amount of displacement along the path. For the $x$ component, a new ray is cast from $Cc$; the angle between this ray and the previously casted ray is the amount of rotation to apply in order to have the picked pixel to stay under the cursor. A control-display gain (cd gain) can be further applied on the angle to allow larger rotation with a reduced hand movement. A value between 1 and 2 seems good candidate.
- on cursor up (termination).

The simplest solution for the walking path needed by Drag'n Go is to use a straight line between source and target. This solution has the advantage of being very simple to implement and also seems to provide good user feedback. A straight line is a good solution for large empty space, but we expect that more sophisticated methods for path calculation could be used to improve displacement aesthetics or more realistic movements.

## 4 Experiment

To evaluate and improve the Drag'n Go technique we conducted a pilot experiment in which we evaluate learning time, ease of use of the method and its efficiency, compared to other direct navigation techniques. We also wanted to gain insight into the impact of the input device on the technique performance.

### 4.1 Method

The pilot experiment consists in a long loop corridor with corners (see Figure 3a) that the user has to cross as fast as possible. Green arrows placed on walls and ground indicate the direction to follow (see Figure 3b). The experiment has two conditions, according to the input device: (1) keyboard and/or mouse and (2) multitouch screen. In condition 1, we compare three techniques: a) a FPS navigation where arrow keys control the camera position and the mouse controls the camera orientation, b) the POI technique as described by Mackinlay [10] and c) Drag'n Go with a straight-line path. In condition 2, we compared Drag'n go with two techniques: a) a local navigation in which one finger controls the pan and dolly in/out movement and two fingers control the tilt and sidestep movement, inspired from [5], b) an adaptation of POI for touch screens.

### 4.2 Apparatus and Procedure

Our experimental setup consists in: standard keyboard and mouse, and a 22-inch 3M multi-touch screen with a 6 ms latency. Eight subjects participated in this experiment (8 males, average age = 25),

(a) top view (b) screenshot

Figure 3: Scene of experiment.

all of them are frequent users of computers and video games. The techniques appear with a randomized ordering pattern, and at each change of technique a training session is proposed. During this session, a caption (see the right-bottom corner in Figure 3b) informs the subject about which device and input DoFs they can use but not how this input DoF are mapped to the camera DoFs. At the end of a training session, the subject can continue practicing or start the real task. Each condition is composed of three blocks. The duration of each repetition as well as the duration of the training session are recorded. After the last block the subjects are invited to fill in a questionnaire with Likert scales. Five questions are asked in both conditions:

1. Was the interaction technique *easy to understand* ?
2. Was the interaction technique *easy to use* ?
3. Did you perceive interaction as *free* ?
4. Did you perceive interaction as *accurate* ?
5. Did you perceive interaction as *fast*?

To each question, the subject rates his agreement between 0 to 6.

### 4.3 Results

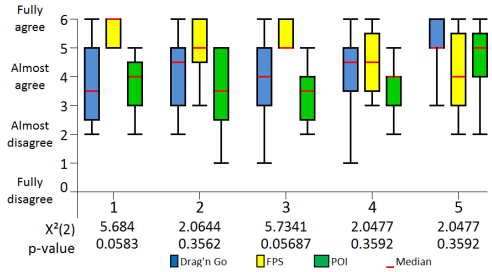
Quantitative data resulting from the measurement are in Table 3. Our quantitative datum fails to pass normality and homogeneity tests (Shapiro and Bartlett tests); consequently we used a non-parametric method, the Friedman's test, to evaluate the significance of means differences. Despite being less powerful, Friedman's test already reports significant effects. In the touch screen condition, the choice of a technique has a significant impact on completion time ($\chi^2(2) = 8.2$, *p=0.016)*, as well as a significant difference in the training time ($\chi^2(2) = 6.2$, *p=0.04)*. Post-hoc analysis is done using a pairwise comparison of Wilcoxon rank test. On the task completion time the means difference is explained by Drag'n Go vs Local (*p=0.065*) and POI vs local (*p=0.082*) more than Drag'n Go vs POI (*p=1.0*). For the training time, difference is explained by POI vs Local (*p=0.065*), more than Drag'n Go vs Local (*p=0.232*) and POI vs Drag'n Go (*p=1.0*). Moreover, in Table 3, we see that the local navigation required more time than other techniques and Drag'n Go has a short training time and a good efficiency.

In the keyboard+mouse condition there is a significant difference between means for the training time ($\chi^2(2) = 7$, *p=0.03)* but not for the task completion time ($\chi^2(2) = 3$, *p=0.22)*. It results from this that studied techniques do not require similar learning time but are equivalent for task completion. Post-hoc analysis of the training time shows that the difference between FPS vs POI (*p=0.022*) and Drag'n Go vs POI (*p=0.054*).

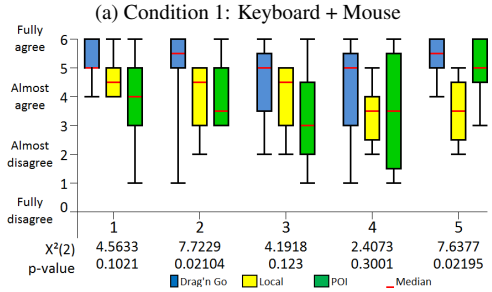
Based on answers to questionnaires (see Figure 4), in the keyboard+mouse condition, we noticed that the FPS navigation is the most appreciated technique but there are no significant differences for a *p-value=0.05*. In touch screen condition, the choice of a technique has a significant impact on the ease to use ($\chi^2(2) = 7.7$, *p-value=0.021*) and the perception of a fast navigation ($\chi^2(2) = 7.6$, *p-value=0.021*). For other questions, ranks of Drag'n Go are high showing that the subjects appreciate our technique.

| Navigation technique | Tr. time | Cmpl. time |
|---|---|---|
| FPS (keyboard + mouse) | *31s, 24s, 17* | *14s,14s,1* |
| POI (keyboard + mouse) | *67s, 63s, 34* | *14s,14s,3* |
| Drag'n go (keyboard + mouse) | *30s,27s, 12* | *16s,16s,3* |
| POI (touch screen) | *36s,29s,25* | *15s,12s,5* |
| Local (touch screen) | *67s,71s,22* | *22s,16s,10* |
| Drag'n go (touch screen) | *42s,45s,11* | *14s,13s, 4* |

Table 3: Means, medians and standard deviations for both training and completion time for different devices and navigation techniques.




(a) Condition 1: Keyboard + Mouse




(b) Condition 2: Touch screen

Figure 4: User satisfaction ranks for the experiment. The chi-squared value and p-value are indicated under each question number.

In addition to quantitative and qualitative results, we noticed that some subjects were not initially able to distinguish between Drag'n Go and POI in both conditions.

## 5 Discussion

The analysis of the questionnaires shows that the general usability of Drag'n go is good. The subjects do not have difficulties when using the technique and the Drag'n go technique seems to be the preferred solution on the touch screen, as well as a competitive technique when using the keyboard and mouse (see Figure 4). This is confirmed by the significant difference in the training time. Experiment carried out may need to be enriched with more subjects to increase result significance. It is also important to notice that the tested subjects were experienced users of 3d softwares. And thus there is a possible bias, in the proposed experiment results, in favor of FPS navigation. Having a second group of novice subjects would add important insight to the behavior of the Drag'n Go technique. We also think that the conducted experiment is based on a very specific task and that it may not reflect all the properties of the Drag'n Go technique. For example, with Drag'n Go it is possible to control the traveling speed and the final position, which is not possible with other context base methods like POI or Navidget. The results obtained by the Drag'n Go against POI, FPS and Local navigation are promising but do not allow to draw general conclusions about the technique. For example it would be interest-

ing to experiment how the method would perform against indirect techniques like Navidget [7] (Our experiment uses a questionnaire inspired from [7]), against The Follow my Finger technique and against the multi-scale approach of [11].

Doing such experiment is also very useful as it allows us to observe how the users discover and perceive the interaction techniques. We observe that some subjects mimick the Drag'n Go technique while working with POI and still do so when informed it is useless. By doing so they make the two techniques look very similar. With Drag'n Go the user's movement is equivalent to a pointing action following the Fitt's law and thus a movement similar to the one generated programatically by the POI approach. With POI the control of the distance to the target is done through duration. A possible explanation as to why users continue to mimic Drag'n Go is that this gesture may be a way to self-measure efficiency through perception of interaction duration. Nevertheless we see this as an indirect confirmation that the Drag'n Go is appreciated by the users as they even try to give to POI the same screen-space flavor. Some subjects also commented that FPS navigation is less demanding in terms of cognitive load and that they can navigate without thinking about it. Of course this is strongly dependent on how experienced the subjects are.It may also be partially explained by the fact that the other techniques require to regularly clutch the cursor to move forward. This may have an impact on the user's fatigue and it may also force the user to remain focused on the navigation task.

We also observed user experiencing a two-finger gesture to turn in a similar way to the one used to rotate photos on mobile phones. We further investigated user intend on this point, to know if the subjects wanted to turn around themselves or if they preferred to orbit the camera. The answer given depends on the situation, and movement expected by user seems to be correlated with the location of the finger with minimal movement. In the case where that finger is located near the bottom-center of the screen the user wants to rotate around himself; in all other cases, point of view shall turn around the picked object.

## 6 Conclusion and Future work

We proposed a new navigation technique called Drag'n Go. The main idea of Drag'n Go is to provide a direct way to control the camera in multi-scale virtual environments. The method is simple to implement, requires a reduced amount of input degree of freedom and can be used on a large range of devices. We conducted pilot experiment revealing that Drag'n Go is an efficient and appreciated method.

One possible improvement to the presented method would be to handle additional movement (as orbiting) to facilitate inspection tasks. Doing such on touch screen should probably follow the two-finger rotation gesture done by the subjects of the experiment. On other device it is still unclear what should be the simplest and most direct gesture to control of the orbit.

A second improvement would be to add a navigation mode which requires less attention from the user in a way similar to the fly mode [11]. It should be possible, if the cursor stays static to make the avatar continues his travel at a constant speed. Another possible way to improve the proposed technique would be to propose some other path models (following [3] or [12]). Nevertheless the practical impact of the path on the navigation task is not obvious yet, and it is an opened question to experiment the impact of path shape on navigation.

## References


[1] N. Burtnyk, A. Khan, G. Fitzmaurice, R. Balakrishnan, and G. Kurtenbach. Stylecam: interactive stylized 3d navigation using integrated spatial & temporal controls. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, UIST '02, pages 101–110, New York, NY, USA, 2002. ACM.

[2] G. Casiez, D. Vogel, Q. Pan, and C. Chaillou. Rubberedge: reducing clutching by combining position and rate control with elastic feedback. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 129–138, New York, NY, USA, 2007. ACM.

[3] M. Christie and E. Languénou. A constraint-based approach to camera path planning. In *Proceedings of the 3rd international conference on Smart graphics*, SG'03, pages 172–181, Berlin, Heidelberg, 2003. Springer-Verlag.

[4] M. Christie and P. Olivier. Camera control in computer graphics: models, techniques and applications. In *SIGGRAPH ASIA '09: ACM SIGGRAPH ASIA 2009 Courses*, pages 1–197, New York, NY, USA, 2009. ACM.

[5] J. Edelmann, A. Schilling, and S. Fleck. The dabr - a multitouch system for intuitive 3d scene navigation. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2009*, pages 1 –4, may 2009.

[6] M. Gleicher and A. Witkin. Through-the-lens camera control. *SIGGRAPH Comput. Graph.*, 26:331–340, July 1992.

[7] M. Hachet, F. Decle, S. Knodel, and P. Guitton. Navidget for easy 3d camera positioning from 2d inputs. In *3D User Interfaces, 2008. 3DUI 2008. IEEE Symposium on*, pages 83 –89, march 2008.

[8] T. Igarashi, R. Kadobayashi, K. Mase, and H. Tanaka. Path drawing for 3d walkthrough. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, UIST '98, pages 173–174, New York, NY, USA, 1998. ACM.

[9] A. Khan, B. Komalo, J. Stam, G. Fitzmaurice, and G. Kurtenbach. Hovercam: interactive 3d navigation for proximal object inspection. *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 1(212):73–80, 2005.

[10] J. D. Mackinlay, S. K. Card, and G. G. Robertson. Rapid controlled movement through a virtual 3d workspace. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '90, pages 171–176, New York, NY, USA, 1990. ACM.

[11] J. McCrae, I. Mordatch, M. Glueck, and A. Khan. Multiscale 3d navigation. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 7–14, New York, NY, USA, 2009. ACM.

[12] T. Oskam, R. W. Sumner, N. Thuerey, and M. Gross. Visibility transition planning for dynamic camera control. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 55–65, New York, NY, USA, 2009. ACM.

[13] J. L. Reisman, P. L. Davidson, and J. Y. Han. A screen-space formulation for 2d and 3d direct manipulation. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 69–78, New York, NY, USA, 2009. ACM.

[14] B. Shneiderman. The future of interactive systems and the emergence of direct manipulation. In *Proc. of the NYU symposium on user interfaces on Human factors and interactive computer systems*, pages 1–28, Norwood, NJ, USA, 1984. Ablex Publishing Corp.

[15] D. S. Tan, G. G. Robertson, and M. Czerwinski. Exploring 3d navigation: combining speed-coupled flying with orbiting. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '01, pages 418–425, New York, NY, USA, 2001. ACM.

[16] C. Ware and D. Fleet. Context sensitive flying interface. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, I3D '97, pages 127–ff., New York, NY, USA, 1997. ACM.

[17] C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. In *I3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 175–183, New York, NY, USA, 1990. ACM.